UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

CRT-589

DIRECTED STEINER TREE PROBLEM ON A GRAPH: MODELS, RELAXATIONS, AND ALGORITHMS

by

Moshe Dror[1,2]
Bezalel Gavish[3]
Jean Choquette[1]

[1] INRS-Telecommunications, 3 place du Commerce, Verdun, Québec, H3E 1H6, Canada.

[2] Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale A, Montréal, Québec, H3C 3J7, Canada.

[3] Department of Administrative Sciences, Naval Postgraduate School, Monterey, CA 93943, U.S.A.

Cette étude a été publiée grâce à une subvention du fonds F.C.A.R. pour l'aide et le soutien à la recherche.

Université de Montréal
Centre de recherche sur les transports - Publication #589
Août 1988

Dépôt légal - 3e trimestre 1988

ABSTRACT

A Steiner Problem in graphs is the problem of finding a set of edges (arcs) with minimum total weight which connects a given set of nodes in an edge-weighted graph (directed or undirected). This paper develops models for the directed Steiner tree problem on graphs. New and old models are examined in terms of their amenability to solution schemes based on Lagrangean relaxation. As a result, three algorithms are presented and their performance compared on a number of problems originally tested by Beasley (1984, 1987) in the case of undirected graphs.

Key words: Steiner problem; Lagrangean relaxation.

RÉSUMÉ

Étant donné un graphe G = (V,A) et un sous-ensemble de sommets V_1 de V, le problème de Steiner consiste à déterminer un graphe partiel de G de longueur minimale permettant de relier entre eux tous les sommets de V_1 en utilisant éventuellement un ou plusieurs sommets de $V \backslash V_1$. Le présent article traite de ce problème dans le cas des graphes non orientés. Différentes modélisations y sont examinées, ainsi que des méthodes de solution basées sur la technique de la relaxation Lagrangienne. En guise de résultats, trois algorithmes sont présentés et testés sur un ensemble de problèmes originalement utilisés par Beasley (1984, 1987) dans le cas des graphes non orientés.

Mots-clés : problème de Steiner; relaxation lagrangienne.

1. INTRODUCTION

1.1 Graph terminology

A directed graph $G = \{V, A\}$ consists of a finite set of vertices V ($|V| = n$) and a set of arcs A ($|A| = m$). Each arc in A is an ordered pair (v_i, v_j) of distinct vertices v_i, v_j ($A \subset V \times V$). A weighted graph $G = \{V, A; g\}$, is the graph G together with a nonnegative real function g defined over A. Since our concern is only with weighted graphs, we drop the function g from our notation for graphs unless otherwise specified.

A subgraph $G' \subseteq G$ is a graph $G' = \{V', A'\}$, where $V' \subseteq V$ and $A' \subseteq A$ and such that each arc a'_k of G' ($a'_k \in A'$) is incident to the same vertices in G' and in G. The weight of a subgraph G', denoted by $W(G')$, is the sum of the weights c_k (or c_{ij} for $a_k = (v_i, v_j)$) over all the arcs a_k in A'.

A *semipath* joining vertices v_s and v_t in G is a sequence of vertices $v_s, v_{s+1}, ..., v_t$ such that the arc $(v_{s+i}, v_{s+i+1}) \in A$ or $(v_{s+i+1}, v_{s+i}) \in A$ for each i. A set of vertices $U \subseteq V$ is *weakly connected* if there is a semipath of vertices in U joining any pair of vertices in U. A *path* from v_s to v_t in G is a sequence of arcs in A such that $(v_s, v_{s+1}), (v_{s+1}, v_{s+2}), ..., (v_{s+i}, v_{s+i+1}), ..., (v_{t-1}, v_t)$. The path is *simple* if all the vertices $\{v_s, v_{s+1}, ..., v_t\}$ are distinct. A path is a *cycle* if $v_s = v_t$. A set of vertices $S \subseteq V$ is *strongly connected* if there is a path from any vertex in S to any other vertex in S which does not contain any vertices in $V \setminus S$. A maximal strongly connected set is a *strongly connected component* of G. An *arborescence* B, ($B \subseteq G$) is a set of arcs such that

(i) if $(v_i, w_i), (v_j, w_j)$ are distinct arcs of B, then $w_i \neq w_j$; (at most one arc entering a vertex)

(ii) B does not contain a cycle;

(iii) B is weakly connected. (B is a directed tree.)

A graph $G = \{V, A\}$ is said to be *connected* if the set V is weakly connected. Let $G' = \{V', A'\}$ be a subgraph of $G = \{V, A\}$ and $V'' \subseteq V'$, then G' is said to be a *connected subgraph with respect to* V'' if there exists a semipath in G' between each pair of distinct vertices in V''. Furthermore, if G' is an aborescence, then subgraph G' is said to be an *aborescence subgraph* of G with respect to V'', and if $V' = V'' = V$ then G' is a *spanning arborescence* of G. Note that one vertex in V' is designated as the *root* for the aborescence.

1.2 Problem Statement

Let $G = \{V, A\}$ be a weighted connected graph. Let $V_1 \subseteq V$ and $G(V_1) = \{G_i \subseteq G;$ such that G_i is a connected subgraph with respect to $V_1\}$. The problem is to find the least weight graph in $G(V_1)$. Such graph (denoted by $G^*(V_1)$) is called a *directed Steiner graph of G with respect to V_1*. If $G^*(V_1)$ is an aborescence then it is called a

directed Steiner tree of G with respect to V_1 or the *minimal arborescence of G with respect to* V_1. The vertices of the graph $G^*(V_1)$ not from the set V_1, are referred to as *Steiner points.*

Our graph notation attempts to tie together the notational conventions developed by Bondy and Murty (1976), Hakimi (1971) and Tarjan (1977). An arc from v_i to v_j is denoted either by (v_i, v_j) or simply by (i, j) if no ambiguity results.

We restrict the discussion to weighted graphs with strictly positive weight functions and to the problem of finding the minimal arborescence of G with respect to V_1, i.e., the directed Steiner tree problem. Any solution to this problem, designates one vertex in V_1 as the root vertex of the resulting minimal arborescence of G with respect to V_1. To facilitate the forthcomming mathematical formulations, we augment the graph G with an additional vertex denoted as vertex 0 and with a set of arcs $(0, v_i)$, for $v_i \in V_1$. The weight function g assigns to these arcs high positive identical weights to assure only one such arc in the optimal solution to the directed Steiner tree problem.

The complexity status of the Steiner tree problem on graphs is well settled. Karp, 1972, proves the NP-completeness of this problem by polynomial transformation from Exact Cover by 3-Sets problem. The problem remains NP-complete if all the arc weights are equal, if G is a bipartite graph with no arcs joining two vertices in V_1 (or two vertices in $V \setminus V_1$), and also in case G is a planar graph. For more detailed reference list on the complexity of this problem see the classic book by Garey and Johnson, 1979. As far as a literature review of past research on the Steiner tree problem on graphs, the readers are directed to the very fine recent paper by Winter, 1987.

2. MATHEMATICAL FORMULATIONS WITH FLOW VARIABLES

In the mathematical formulations that follow, the decision variables are of two basic types; (i) a binary (selection) variables - y_{ij}, which accept value 1 if the arc (v_i, v_j) is selected in the solution and the value 0 if not, and (ii) flow variables either x_{ijp} (or x_{ij}) representing the amount of flow through the arc (v_i, v_j) directed from the root vertex 0 to vertex p in V_1 (or the total flow on the arc (v_i, v_j)).

2.1 Mathematical Formulation - A

Minimize $z = \sum_{(i,j)\in A} c_{ij} y_{ij}$, (1)

subject to:

$$\sum_{j\in V\cup\{0\}} x_{ijp} - \sum_{j\in V\cup\{0\}} x_{jip} = \left\{\begin{array}{l} 1, \text{ for } i = 0, \\ 0, \text{ for } i \neq 0, p \\ -1, \text{ for } i = p, \end{array}\right\} \text{ for all } p \in V_1 \qquad (2)$$

$$x_{ijp} \leq y_{ij}, \qquad \text{for all } (v_i, v_j) \in A,\ p \in V_1 \qquad (3)$$

$$x_{ijp} \geq 0, \qquad \text{for all } (v_i, v_j) \in A, p \in V_1 \qquad (4)$$

$$y_{ij} = 0 \text{ or } 1, \qquad \text{for all } (v_i, v_j) \in A \qquad (5)$$

Given the objective of minimizing the total weight of the selected arcs (the arcs for which $y_{ij} = 1$), the constraints have to ensure the arborescence structure (with respect to V_1). Constraints (2) impose the conservation of flow on all vertices in $V \setminus (V_1 \cup \{0\})$ and ensure that one unit of flow leaves the root vertex 0 for each destination vertex p in V_1. Constraints (2) also state that one unit of flow reaches each vertex in V_1. Thus, there has to be a path (in G) from the vertex 0 to each one of the vertices in V_1. Constraints (3) ensure that flow is allowed only through the arcs selected in the solution. Constraints (4) are the flow nonnegativity constraints and (5) are the binary value constraints for the appropriate decision variables.

A note: In case the weight function g of the graph $G = \{V, A; g\}$ is not strictly positive, then in order to ensure a tree structure for the solution to the MDSTP we add the constrains $\sum_{i\in V} y_{ij} = 1$ for all $j \in V_1$ (2a). Since in this paper we restrict the discussion to strictly positive functions g, constraints (2a) are redundent and subsequently dropped.

A more compact mathematical formulation to the one presented in **(A)** is obtained by simply aggregating constraints (3) with respect to the index p. In this case we obtain:

$$\sum_{p\in V_1} x_{ijp} \leq |V_1| y_{ij}, \qquad \text{for all } (v_i, v_j) \in A \qquad (3')$$

We denote the mathematical formulation which contains the equations (1), (2), (3'), (4), and (5) as **(A')**.

It is of interest to examine the implications of aggregating constraints (3) into the form of (3'). Since both formulations construct the same weight solution if solved optimaly, what could be the advantage of attempting to solve one set of equations versus the other set of equations ? The answer to this question is in most cases experimental in the sense that a number of researchers (Magnanti and Wong, 1981, Cornuejols et al., 1977) were much more successfull in constructing efficient solution schemes for the disaggregated formulation **(A)** than the more compact **(A')** formulation. In order to formalize this experimental experience we examine the linear programming relaxations (constraints (5) become ≥ 0, and $y_{ij} \leq 1$ for all $(v_i, v_j) \in A$) of both

formulations. Denote the objective function value for the LP relaxation for **(A)** as z^{A}_{LP} and for **(A')** as $z^{A'}_{LP}$.

Theorem 1 : $z^{A}_{LP} \geq z^{A'}_{LP}$.

Proof : It is clear that any feasible solution for the linear programming relaxation of **(A)** is also feasible for the linear programming relaxation of **(A')**. In the reverse direction this is not true in general. One can illustrate it by examining a equilateral triangle graph with 4 vertices, (one in the center) and 7 directed arcs (see Figure 1). The V_1 set consists of vertices 1 and 2.

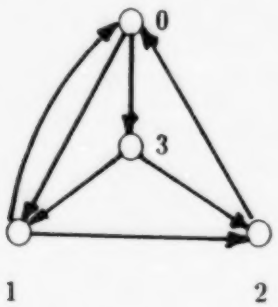

Figure 1 : Equilateral triangle graph.

The flow variables $x_{031} = 2, x_{311} = 2, x_{121} = 1, x_{201} = 1$ are clearly feasible in the LP relaxation for **(A')** and not feasible for the LP relaxation for **(A)** •

Given two (different) equivalent integer programming formulations (P1) and (P2) for an optimization (minimization) problem. Denote by (RP1) and (RP2) the respective formulations obtained by relaxing the integrality constraints. If $z_{RP1} \geq z_{RP2}$ and not equal in general, then formulation (P1) is called a **strong formulation** with respect to (P2). This concept of strong formulation is stated in Geoffrion, 1979 and Van Roy, 1986.

In our case, we proved that **(A)** is a strong formulation with respect to **(A')**.

The next mathematical formulation for the minimal arborescence problem with respect to V_1 requires only two indices for its flow variables. It is based on a similar TSP formulation from Gavish and Graves, 1982.

Mathematical Formulation - B

Minimize $z = \sum_{(i,j)\in A} c_{ij} y_{ij}$, (6)

Subject to:

$$\sum_{j=1}^{n} x_{kj} - \sum_{i=0}^{n} x_{ik} = -1, \qquad \text{for all } k \in V_1 \tag{7}$$

$$\sum_{j=1}^{n} x_{kj} - \sum_{i=0}^{n} x_{ik} = 0, \qquad \text{for all } k \in V \setminus V_1 \tag{8}$$

$$x_{ij} \leq |V_1| y_{ij}, \qquad \text{for all } (v_i, v_j) \in A \tag{9}$$

$$x_{ij} \geq 0, \qquad \text{for all } (v_i, v_j) \in A \tag{10}$$

$$y_{ij} = 0 \text{ or } 1 \qquad \text{for all } (v_i, v_j) \in A \tag{11}$$

The formulations **(A)** and **(B)** are equivalent in terms of determining the same optimal solution. In terms of the linear programming relaxation, formulation **(A)** is a strong formulation with respect to **(B)** (the proof of Theorem 1 holds for this case too). What is also clear is that any feasible solution (in terms of flow) of the LP relaxation for **(A')** is also feasible for the LP relaxation of **(B)** and visa versa. Thus, **(A')** and **(B)** are fully equivalent.

3. MATHEMATICAL FORMULATIONS WITHOUT FLOW VARIABLES

3.1 A Note on the Minimal Spanning Arborescence Problem - (MSAP)

When attempting to solve the MDSTP on a graph, one usually reexamines the very similar 'easy' problem of constructing a minimal cost spanning arborescence (MSAP). In all the formulations of MSAP which are known to the authors, flow variables are used (see for example Gavish, 1982) in the same fashion as in the MDSTP formulations presented above in Section 2.1. Aneja, 1980, presents a formulation of the MDSTP without the flow variables. His is a set covering formulation in which the number of constraints grows exponentially with the size of problem instances. We present a very simple formulation of the MSAP which does not require flow variables nor does the number of constraints grow exponentially with the size of problem instances. In this new MSAP formulation we 'borrow' a version of TSP subtour elimination constraints (Miller, Tucker, and Zemlin, 1960).

Minimize $z = \sum_{(i,j) \in A} c_{ij} y_{ij}$

Subject to

$$\sum_{i=0}^{n} y_{ij} = 1 \qquad \text{for all } j = 1, 2, ..., n \tag{12}$$

$$u_i - u_j + n y_{ij} \leq (n-1) \qquad \text{for all } i, j \in V \cup \{0\} \tag{13}$$

$$y_{ij} = 0 \text{ or } 1 \qquad \text{for all } i, j \in V \cup \{0\} \tag{14}$$

where u_i and u_j are arbitrary real numbers.

Note that those cycle (subtour) elimination constraints (14) are for all $i,j \in V \cup \{0\}$ including the 'artificial' root vertex 0 (contrary to $i,j \neq 0$ in the TSP formulation). Also note that the TSP subtour elimination constraints of the type $\sum_{i\in S}\sum_{j\in\overline{S}} y_{ij} \leq |S| - 1$ and $\sum_{i\in S}\sum_{j\in\overline{S}} y_{ij} \geq 1$ for all $S \subseteq V$ would not be appropriate for this MSAP formulation.

3.2 Set Covering Formulation - C

First, we present a set covering type formulation for the MDSTP, modified for the directed graph by Wong, 1984, and originally presented by Aneja, 1980, for the undirected graph.

Minimize $z = \sum_{(i,j)\in A} c_{ij}y_{ij}$ (15)

Subject to:

$\sum_{(i,j)\in A, i\in N_1, j\in\overline{N}_1} y_{ij} \geq 1$ for all $N_1 \subseteq V$ such that $0 \in N_1$ and $\overline{N}_1 \cap V_1 \neq \emptyset$ (16)

($\overline{N}_1$ denotes the complement of N_1 in V)

$$y_{ij} = 0 \text{ or } 1 \qquad \text{for all } (i,j) \in A \qquad (17)$$

In his solution scheme for formulation **C**, Aneja, 1980, solves optimaly the linear programming relaxation of **C**. An interesting result due to Wong, 1984, is that $z^A_{LP} = z^C_{LP} (\geq z^{A'}_{LP} = z^B_{LP})$. (I.e., the linear programming relaxations of the set covering formulation **C** and the flow formulation **A**, have the same optimal values.)

The number of constraints in formulation **C** is exponential in the size of the problem. There is too little 'structure' in this formulation to be usefull in Lagrangean relaxation schemes. In order to amend this 'weakeness' we add a number of redundent structural constraints in the next formulation.

3.3 Modified Set Covering Formulation - D

Minimize $z = \sum_{(i,j)\in A} c_{ij}y_{ij}$ (18)

Subject to:

$$\sum_{i=0}^{n} y_{ij} = 1 \qquad \text{for all } j \in V_1 \qquad (19)$$

$$\sum_{(i,j)\in A} y_{ij} \geq |V_c| - 1 \qquad (20)$$

$$u_i - u_j + n y_{ij} \leq (n-1) \qquad \text{for all } (i,j) \in A \qquad (21)$$

$$\sum_{(i,j)\in C(p)} y_{ij} \geq 1 \qquad \text{for all } p \in V_1 \text{ and all cuts } \{C(p)\} \qquad (22)$$

where $C(p)$ is a cut (a subset of arcs in A) between the vertex 0 and the vertex $p \in V_1$

$$y_{ij} = 0 \text{ or } 1 \qquad \text{for all } (i,j) \in A \qquad (23)$$

u_i and u_j are arbitrary real numbers.

The objective function (18) and the constraints (22), (23) define the MDSTP in a formulation equivalent to **C**. Constraints (19), (20), and (21) are redundent. The constraints in (19) ensure that only one arc enters each vertex in V_1. Constraints (20) state that at least $|V_c| - 1$ arcs are selected in any solution where initially $V_c = V_1$ and in principle $V_1 \subseteq V_c$. Constraints (21) are the subtour elimination constraints. Constraints (21) **are not** the 'complicating' constraints in this case. The complicating constraints are the set covering constraints (22) which ensure a tree structure solution which spans all the vertices in V_1. Dropping the constraints (22) results, through the (21) constraints in a minimal weight forest solution to the remaining problem.

Corollary 1: $z^D_{LP} \geq z^C_{LP} (= z^A_{LP})$.

This result simply follows from the fact that we have added constraints (19) to an equivalent formulation to **C**. The other constraints types ((20) and (21)) would be satisfied in the **C** formulation through constraints (16).

4. LAGRANGEAN RELAXATION

In this section we describe a number of Lagrangean relaxations for the mathematical formulations presented in Sections 2 and 3. Lagrangean relaxation approach for solving 'hard' problems is based on the observation that by removing the complicating constraints from a mathematical formulation, the resulting problem is 'easily' solvable. A solution to the relaxed problem constitutes a lower bound on the solution to the original problem. The thrust in such an approach is to obtain a maximal lower bound which, if it does not solve the original problem, can be integrated into an implicit enumeration scheme such as branch and bound.

4.1 Lagrangean Relaxations of (A)

We present two Lagrangean relaxations of formulation **A**. In the first one the arc selection constraints are relaxed, resulting in a shortest path type problem. In the second relaxation the conservation of flow constraints are moved into the objective leading to a more difficult subproblem.

4.1.1 The First Relaxation

In formulation **(A)**, the complicating constraints are the arc selection constraints (3), which ensure that the unit flow from the root vertex 0 to a vertex $p, p \in V_1$ passes only through the arcs selected in the solution. Below we present the Lagrangean relaxation obtained by moving the constraints (3) multiplied by nonnegative λ_{ijp} into the objective function. For a given vector of multipliers λ, the problem is

Minimize $z(\lambda) = \sum_{(i,j)\in A} c_{ij}y_{ij} - \sum_{(i,j)\in A} \sum_{p\in V_1} \lambda_{ijp}(y_{ij} - x_{ijp})$

which after rearrangement of terms has the following form

$$\text{Minimize } z(\lambda) = \sum_{(i,j)\in A}(c_{ij} - \sum_{p\in V_1} \lambda_{ijp})y_{ij} + \sum_{(i,j)\in A} \sum_{p\in V_1} \lambda_{ijp}x_{ijp} \quad (24)$$

Subject to:

$$\sum_{j\in V\cup\{0\}} x_{ijp} - \sum_{j\in V\cup\{0\}} x_{jip} = \left\{\begin{array}{l} 1, \text{ for } i = 0, \\ 0, \text{ for } i \neq 0, p \\ -1, \text{ for } i = p, \end{array}\right\} \text{ for all } p \in V_1 \quad (25)$$

$$0 \leq x_{ijp} \leq 1, \qquad \text{for all } (i,j) \in A, p \in V_1 \quad (26)$$

$$y_{ij} = 0 \text{ or } 1, \qquad \text{for all } (i,j) \in A \quad (27)$$

The 'best' Lagrangean value is obtained by maximizing $z(\lambda)$ over nonnegative $\lambda's$ ($\lambda_{ijp} \geq 0$, for all $(i,j) \in A, p \in V_1$). For those optimal Lagrangean multipliers (λ_{ijp}) the following relationship holds (see Gavish, 1978, and Wong, 1984, for the dual formulation of **A**):

$$c_{ij} - \sum_{p\in V_1} \lambda_{ijp} \geq 0 \qquad \text{for all } (i,j) \in A \quad (28)$$

By observing that the above ((24) - (27)) Lagrangean formulation has the Integrality Property (Geoffrion, 1974), the lower bound value obtained for the MDSTP by solving (24) - (27) is equal to the value for the linear programming relaxation to the problem. In this case, the main advantage for examining the Lagrangean relaxation of **A** would depend on how fast, in comparison, can such a relaxation be solved. In addition, such a solution could be more amenable for developing good heuristics.

The (28) inequalities suggest a fast solution procedure to obtain the maximal $z(\lambda)$ solution for the Lagrangean relaxation of **A** based on the repeated use of the shortest path algorithm with cost modifications along the way. In the algorithm outlined below, we successively adjust the values of the λ_{ijp}'s using the subgradient method described in Held et al. (1974) while preserving the dual feasibility of these multipliers via the (28) inequalities. We enforce for all $(i,j) \in A$ throughout the algorithm below (in each modification of multipliers) the following constraints : $\sum_{p\in V_1} \lambda_{ijp} = c_{ij}$. Then the objective of the Lagrangean problem is equivalent to a shortest path problem.

Denote by $z(LA1(p))$ the value of $z(LA1)$ obtained by the Algorithm LA1 outlined below, given that $p, (p \in V_1)$ is the root node of the Steiner tree. Denote by $\bar{z}(LA1(p))$ the weight of the tree generated by $z(LA1(p))$ solution (i.e., assign the actual costs c_{ij} to the arcs in the Steiner tree). Let $s_p(l,p)$ denote the shortest path from l to p in the network with arc costs $(\lambda_{ijp}), (i,j) \in A$.

Algorithm LA1

Step 0 : (Initialization)

Set $\lambda_{ijp} = \frac{c_{ij}}{|V_1|}$ for all $(i,j) \in A, p \in V_1$

$z(\lambda)_{best} = 0; z_{best} = \infty$; Iter = 0, ImpIter = 0;

Idivide = (preset parameter); $\delta = 2$; Iterlim = Maximum number of iterations; ϵ_δ = smaller gap on δ; ϵ_{best} = relative precision desired on $z(\lambda)_{best}$; $st = 0$

Step 1 : Solve the following (Lagrangean relaxation) problem:

min $z_1 = \sum_{p \in V_1} \sum_{(i,j) \in A} \lambda_{ijp} x_{ijp}$ subject to constraints (26) and (27).

This problem can be solved with a shortest path algorithm as follows:

For each $p \in V_1$ compute all the shortest paths from p to $k \in V_1$ using arc costs $\{\lambda_{ijp}\}$ and denote its cost by S_p. Select the solution with the minimal S_p value.

Step 2 : Updating the bounds

Lower bound :

If $z(\lambda) > z(\lambda)_{best}$ *then*

$z(\lambda)_{best} = z(\lambda)$, *and for all* $(i,j) \in A, p \in V_1, \lambda_{ijp}(best) = \lambda_{ijp}, \gamma_{ijp}(best) = \gamma_{ijp}, st(best) = st$, ImpIter = 0

Upper bound :

Consider the subgraph $G' = (V \cup \{0\}, A')$ where $A' = \{(i,j); x_{ijp} > 0$ for some $p \in V_1\}$. Obtain the shortest path tree structure for this subgraph by computing the shortest paths from 0 to each node of V_1. Let $\{y_{ij} = 1\}$ for each arc (i,j) in this shortest path tree and 0 otherwise.

The current solution vector (y,x) is always feasible for the original problem and its cost $z_f(y,x) = \sum_{(i,j) \in A} c_{ij} y_{ij}$

If $z_f(y,x) < z_{best}$ *then*

$z_{best} = z_f(y,x)$, ImpIter = 0

Step 3: Updating λ

(i) *If* ImpIter = Idivide *then*

Change the value δ and restart from the best solution $z(\lambda)_{best}$ (i.e., $\delta = \delta/2$, ImpIter = 0, $z(\lambda) = z(\lambda)_{best}, st = st(best)/2, \lambda_{ijp} = \lambda_{ijp}(best), \gamma_{ijp} = \gamma_{ijp}(best)$ for all $(i,j) \in A, p \in V_1$).

Else : Compute the *ascent direction* γ and the step st for the current solution.

$$\gamma_{ijp} = x_{ijp} - y_{ij}, \text{ for all } (i,j) \in A, p \in V_1$$

$$st = \frac{\delta(z_{best} - z(\lambda))}{||\gamma||^2}$$

(ii) $\bar{\lambda}_{ijp} = \lambda_{ijp} + st\gamma_{ijp}, \quad (i,j) \in A, p \in V_1$

The new multipliers λ_{ijp} are obtained by solving the following problem:

$\min\{||\lambda - \bar{\lambda}||; \sum_{p \in V_1} \lambda_{ijp} = c_{ij}, \lambda_{ijp} \geq 0, (i,j) \in A, p \in V_1\}$

The vector λ is the projection of $\bar{\lambda}$ on $\{\sum_{p \in V_1} \lambda_{ijp} = c_{ij}, \lambda_{ijp} \geq 0, (i,j) \in A, p \in V_1\}$

Note : The solution of the projection problem follows the procedure suggested by Held et al. (1974, pp. 77).

Step 4 : Stoping Conditions

(a) If $Iter \geq Iterlim$ then Stop

(b) If $\delta < \epsilon_\delta$ then Stop

(c) If $\frac{z_{best} - z(\lambda)_{best}}{z(\lambda)_{best}} < \epsilon_{best}$ then Stop

Otherwise Go To Step 2

4.1.2 The Second Relaxation

Following a different relaxation approach, we dualize on the flow constraints (2) using λ_{ip} as the Lagrangean multipliers. The number of multipliers reduces to $n|V_1|$, which is considerably less than in the previous relaxation. The objective function becomes:

Minimize $z(\lambda) = \sum_{(i,j) \in A} c_{ij}y_{ij} - \sum_{p \in V_1} \lambda_{0p}(\sum_{j \in V} x_{0jp} - \sum_{j \in V} x_{j0p} - 1) - \sum_{p \in V_1} \lambda_{pp}(\sum_{j \in V} x_{pjp} - \sum_{j \in V} x_{jpp} + 1) - \sum_{i \in V, i \neq 0,p} \sum_{p \in V_1} \lambda_{ip}(\sum_{j \in V} x_{ijp} - \sum_{j \in V} x_{jip})$

After rearrangement of terms the objective is converted to:

$$\text{Minimize } z(\lambda) = \sum_{(i,j) \in A} c_{ij}y_{ij} + \sum_{p \in V_1} \sum_{(i,j) \in A} x_{ijp}\bar{c}_{ijp} - \sum_{p \in V_1} \bar{c}_{0pp} \quad (30)$$

Subject to (3), (4), and (5) where $\bar{c}_{ijp} = \lambda_{jp} - \lambda_{ip}$ for all $(i,j) \in A$ and $p \in V_1$.

Note that the λ_{ij}'s in this relaxation are unrestricted in sign and the $\bar{c}_{ijp}$ can be handled implicitly storing n^2 entries instead of n^3 entries.

In order to strenghten the lower bound obtained from such a relaxation we amend it with the following constraints:

$$\sum_{i \in V} y_{ij} \geq 1 \quad \text{for all } j \in V_1 \tag{30}$$

$$\text{and} \quad x_{ijj} = y_{ij} \quad \text{for all } (i,j) \in A \text{ and } j \in V_1 \tag{31}$$

The constraints (30), (31) are redundent in the formulation **A**, but are helpful in increasing the Lagrangean bound. Note also that (30) is a relaxation of the (tree) constraint which ensures that only one arc enters a node in V_1 (i.e., $\sum_{i \in V} y_{ij} = 1$ for all $j \in V_1$). The 'tree' constraint is tighter but not easily solvable.

Before presenting a solution procedure for the problem defined by (29), (3), (4), (5), (30), and (31), we make the following observations:

(i) From the selection constraints (3) we notice that:

(a) If $y_{ij} = 0$ then $x_{ijp} = 0$ for all $p \in V_1$

(b) If $y_{ij} = 1$ then

(I) $x_{ijj} = 1$ which follows from (31), and

(II) $x_{ijp} = 1$ if $\bar{c}_{ijp} < 0$, and $x_{ijp} = 0$, if $\bar{c}_{ijp} \geq 0 \wedge p \neq j$.

Note that for the same reason as in the first relaxation of **A**, the maximum lower bound for the MDSTP obtained by solving this relaxation can not exceed the bound obtained from the linear programming relaxation of the problem.

Algorithm LA2

Step 1: (Initialization) Set $\lambda_{ip} = \lambda^0_{ip}$; Iter = 0; ImpIter = 0; $z(\lambda)_{best} = 0$; z_{best} = cost of the best feasible solution found so far; Idivide = preset parameter; $\delta = 2$; st = 0; Iterlim = maximum number of iterations; ϵ_δ = smaller gap on δ; ϵ_{best} = relative precision desired on $z(\lambda)_{best}$.

Note that λ^0_{ip} are randomly generated, z_{best} is obtained by computing the shortest paths from one node of V_1 to all other nodes in V_1.

Step 2: Solve the Lagrangean Problem:

Iter = Iter + 1, ImpIter = ImpIter + 1, Set $y_{ij} = 0, x_{ijp} = 0$, for all $(i,j) \in A, p \in V_1$.

(1) For each arc $(i,j) \in A$

(i) Compute $M_{ij} = c_{ij} + \sum_{p \in V_1, p \neq j} \min\{0; \bar{c}_{ijp}\}$

(ii) If $j \in V_1$ set $M_{ij} = M_{ij} + \bar{c}_{ijj}$

(iii) If $M_{ij} < 0$ then set $y_{ij} = 1$ and

For all $p \in V_1$ set

$$x_{ijp} = \begin{cases} 1 \text{ for } \bar{c}_{ijp} < 0 \text{ or } j = p \\ \\ 0 \text{ otherwise} \end{cases}$$

(2) For each $j \in V_1$, if $M_{ij} \geq 0$ for all $(i,j) \in A$ then determine l such that $M_{lj} = \min_{i;(i,j)\in A}\{M_{ij}\}$ and set $y_{lj} = 1$, for all p such that $\bar{c}_{ljp} < 0$ or $p = j$ set $x_{ljp} = 1$, otherwise set $x_{ijp} = 0$.

(3) Compute $z(\lambda)$

Step 3: (Updating the bounds)

(1) If $z(\lambda) > z(\lambda)_{best}$ then

$$z(\lambda)_{best} = z(\lambda), \lambda_{best} = \lambda, \gamma_{ip}(best) = \gamma_{ip}, \text{st(best)} = \text{st}, ImpIter = 0.$$

(2) If the solution (y,x) is feasible for the original problem then compute its cost and denote the value by z_F.

If $z_F < z_{best}$ then set $z_{best} = z_F$; Set Impiter = 0;

Step 4: (Updating the λ)

(i) If ImpIter = Idivide *then* set $\delta = \frac{\delta}{2}$, ImpIter = 0, $z(\lambda) = z(\lambda)_{best}, st = st(best)/2, \lambda_{ip} = \lambda_{ip}(best), \gamma_{ij} = \gamma_{ip}(best)$ for all $i \in V, p \in V_1$.

else compute the ascent direction γ and the step st

$$\gamma_{ip} = \begin{cases} (\sum_{j\in V} x_{0jp} - \sum_{j\in V} x_{j0p} - 1) & \text{for all } p \in V_1 \\ (\sum_{j\in V} x_{ijp} - \sum_{j\in V} x_{jip}) & \text{for all } p \in V_1, i \neq 0, p \\ (\sum_{j\in V} x_{pjp} - \sum_{j\in V} x_{jpp} + 1) & \text{for all } p \in V_1 \end{cases}$$

$$st = \delta \frac{z_{best} - z(\lambda)}{||\gamma||}$$

(ii) $\lambda_{ip} := \lambda_{ip} - st\gamma_{ip}, i \in V, p \in V_1$

Step 5: (Stoping condition)

(a) If $Iter \geq Iterlimit$ Stop

(b) If $\delta < \epsilon_\delta$ Stop

(c) If $\frac{z_{best} - z(\lambda)_{best}}{z(\lambda)_{best}} < \epsilon_{best}$ Stop

Otherwise Go To Step 2.

4.2 Lagrangean Relaxation of (D)

The complicating constraints in the **D** formulation are the set covering - (22) constraints. The major difficulty is that the number of constraints in (22) is exponential in the size of V_1 set. On the other hand the number of y_{ij} variables is exactly n(n - 1)/2. This implies that in the linear programming relaxation of this formulation, most of the constraints in (22) are nonbinding. The difficulty lies in finding the binding constraints. By removing the constraints (22) and adding them to the objective function multiplied by the appropriate (nonnegative) Lagrangean multipliers we obtain the following relaxed formulation:

$$\text{Minimize } z(\lambda) = \sum_{(i,j)\in A} c_{ij}y_{ij} + \sum_{c_p\in\Pi} \lambda_{c_p}(1 - \sum_{(i,j)\in C(p)} y_{ij})$$

Subject to constraints (19), (20), (21) and (23) where Π is the index set of all cuts c_p between the vertex 0 and the vertices $p \in V_1$.

This formulation can be rewritten as:

$$\text{Minimize } z(\lambda) = \sum_{c_p\in\Pi} \lambda_{c_p} + \sum_{(i,j)\in A} \bar{c}_{ij}y_{ij} \qquad (32)$$

Subject to constraints (19), (20), (21), and (23) where:

$$\bar{c}_{ij} = \begin{cases} c_{ij} - \sum_{c_p\in\Pi} \lambda_{c_p} & \text{for } (i,j) \in C(p) \quad \text{for some } p \in V_1 \\ c_{ij} & \text{for } (i,j) \notin C(p) \end{cases}$$

This last problem ((32), (19), (20), (21), and (23)) for fixed values of λ can be solved in polynomial time. (Again from Lagrangean duality we get $\bar{c}_{ij} \geq 0$ for all $(i,j) \in A$.) For a given λ vector denote this problem by **LD1**.

The question now reduces to one of finding λ^* which maximizes the value of $z(\lambda)$, i.e., $z(\lambda^*) = Max_{\lambda\geq 0}\{z(\lambda)\}$.

Theorem 2 : $z(\lambda^*) \geq z_{LP}^D$.

Proof : This result is based on the observation that the set of constraints (19), (20), (21), and (23) do not have the Integrality Property (Geoffrion, 1974,).

4.2.1 Dual Ascent Procedure for Initial λ_p Values

In this section we describe a dual ascent procedure for computing a lower bound to the undirected version of the MDSTP using the **D** formulation of the problem. Modifing this dual ascent procedure for a general directed graph is left as an algorithmic exercise.

First we explain and provide an outline of the algorithm followed by a small numerical example and conclude with a detailed description of a dual ascent procedure for finding good multipliers to **LD1**.

The algorithm begins by solving the problem LD1 with $V_c = \{0\} \cup V_1$ and all λ values equal to zero. Let L_B be the cost of this solution. The network structure of this solution is that of a 'sparse' forest F of disconnected components. In case of a directed graph, this problem can be solved with a modified Tarjan's algorithm (Tarjan, 1977). We differentiate between two component types. Components which contain at least one node belonging to $V_1 \cup \{0\}$ are denoted by $\Upsilon = \{T_0, T_1, T_2, ...\}$, where T_0 is the component which contains the root node. The second set of components consists of the points S, $S \subseteq V \setminus V_1$ not contained in any of the components in Υ.

At this point we pick a component $T_k, k \neq 0$, and compute the minimal cost of expanding this component. Let $\delta_k = Min_{j \in T_k, i \in V \setminus T_k}\{\bar{c}_{ij}\}$. δ_k is the multiplier value over the cut seperating the nodes in T_k from all the other nodes. The cost matrix is updated to:

$$\bar{c}_{ij} = \begin{cases} \bar{c}_{ij} - \delta_k \text{ for all } j \in T_k, i \in V \setminus T_k \\ \\ \bar{c}_{ij} \qquad \text{otherwise} \end{cases} \tag{33}$$

As a result of cost matrix update, one or more arcs on the cut have a reduced cost of zero.

Let $L_{B_t} = L_{B_{t-1}} + \delta_k$ where t is the iteration number.

The component T_k is merged via the zero cost arcs with a number of other components. This process continues till T_k is merged with T_0 at which point another component in Υ is selected. At each iteration at least one component is added to T_k thus this merging of components stops at most after $|S| + |\Upsilon| - 1$ steps. When the process is completed we have only one component T_0 which contains a subset S_1 of nodes (Steiner points), $S_1 \subseteq V \setminus V_1$. We remove from T_0 all the nodes $s, s \in S_1$ which are dangling nodes (i.e., their degree is ≤ 1). Let S_0 be the set of terminal nodes.

$$L_B = L_{B_f} - \sum_{s \in S_0} c_{p_s,s}$$

where p_s is the adjacent node of node $s, s \in T_0$. This last 'trimming' step is repeated until all the terminal nodes (degree ≤ 1) are nodes in V_1 only.

L_B is the lower bound value to the MDSTP.

In order to illustrate those steps we use the following example. The example consists of 3 required nodes (1, 3, and 4, where node 1 is designated as the root node) and 4 potential Steiner points. The network matrix (the arc weights) is symmetric, which in the directed graph version implies two arcs of the same weight and oposite

direction between two adjacent nodes of the graph. The initial lower bound obtained for this example is 5. The lower bound after two multiplier adjustments is 8, while the optimal solution to the problem is 9.

Example 1 :

	1	2	3	4	5	6	7
1	-	3	5	5	6	6	4
2	3	-	3	3	4	4	6
3	5	3	-	5	1	6	7
4	5	3	5	-	6	1	8
5	6	4	1	6	-	7	8
6	6	4	6	1	7	-	9
7	4	6	7	8	8	9	-

Figure 2 : The original 'distance' matrix.

The initial 'forest' solution connects node 1 to node 2, node 3 to node 5, and node 4 to node 6. This solution has a value of 5. We pick the tree containing the nodes 3 and 5 together with the arc from 3 to 5 as our T_k tree in the algorithm. The corresponding δ_1 value is 3 (the arc weight from node 2 to node 3) and the new lower bound is 5 + 3 = 8. The modified distance matrix is as follows:

	1	2	3	4	5	6	7
1	-	3	2	3	6	4	4
2	3	-	0	1	1	2	6
3	2	0	-	0	1	1	4
4	3	1	0	-	1	1	6
5	3	1	1	1	-	2	5
6	4	2	1	1	2	-	7
7	4	6	4	6	5	7	-

Figure 3 : The modified 'distance' matrix after one descent.

A new tree is selected since the previous one contains the root node 1. The new tree is the arc from node 4 to node 6 together with the two nodes. The new δ_k corresponds to the arc from node 3 to node 4 and has a weight of 2. The new lower bound is 5 + 3 + 2 = 10. Since the expanded new tree contains the root node and there are no trees remaining, we trim the tree from the dangling not required nodes and obtain a tree which contains the nodes 1, 2, 3, and 4 and the arcs (1,2), (2,3),

and (3,4) for the total of 10 - 2 = 8. This solution corresponds to a feasible Steiner tree of value of 11 instead of the optimal value 9.

	1	2	3	4	5	6	7
1	-	3	2	3	6	4	4
2	3	-	0	1	1	2	6
3	2	0	-	0	1	1	4
4	3	1	0	-	1	1	6
5	3	1	1	1	-	2	5
6	4	2	1	1	2	-	7
7	4	6	4	6	5	7	-

Figure 4 : The final modified 'distance' matrix.

4.2.2 The Dual Ascent Algorithm

Step 1 : We start with all $\lambda_{c_p} = 0$ and solve the problem ((32), (19), (20), (21), and (23)) to obtain z(0). The solution to this problem is in the form of a not necessarily connected set of trees Υ which might not include all the nodes in V (i.e., a 'sparse' forest). In case we obtain only one tree then we have the optimal solution for the Steiner tree problem. Denote by L_B the cost of this solution.

Step 2 : Let $|\Upsilon|$ be the number of trees in the forest and let T_k be the set of nodes in tree k. One of these trees contains the root node. Denote that tree by T_0. Denote by S the set of nodes in $V \setminus \cup_{i=0}^{|\Upsilon|} T_k$ (i.e., the potential new Steiner points).

Step 3 : Pick one of the trees in $\Upsilon \setminus T_0$. Tree T_m for example. Compute

$$\Delta = Min_{j \in T_m, i \notin T_m} \{\bar{c}_{ij}\}$$

Let: $L_B = L_B + \Delta$, $\bar{c}_{ij} = \bar{c}_{ij} - \Delta$, for all $i \in T_m, j \notin T_m$, or $j \in T_m, i \notin T_m$

Step 4 : (**Merging**) Every component in $\Upsilon \setminus T_m$ and S with exactly one zero cost arc to T_m (i.e., $\bar{c}_{ij} = 0 \quad j \in T_m, i \notin T_m$) is merged with T_m creating a new forest. (In case of components with multiple zero cost arcs to T_m see **Remark 1**.) Rename the trees in the new forest. Repeat Steps 3 and 4 until T_m is merged with T_0.

Step 5 : If $|\Upsilon| \neq 1$ then go to Step 2, otherwise: eliminate all the terminal nodes in T_0 which belong to $V \setminus V_1$ and reduce the corresponding L_B value by the corresponding arc costs. I.e., $L_B = L_B - \sum_{s \in S_0} c_{p_s,s}$. L_B is the lower bound value for the Steiner tree problem.

Remark 1 : If we add multiple zero cost arcs between T_m and another tree in

T then a cycle is created. Thus, at each merge operation only one arc can be added between any pair of trees. Since without loss of generality we can assume (different) integer arc weights, we can expect that the number of zero cost arcs between a pair of trees in a merge step is small. This suggests a parallel processing algorithm for the construction of the new trees (merged trees) at Step 5. In each case were more than one (say k) zero cost arc exists between a pair of trees, k new merged trees will be stored and processed in parallel. At the end, following Step 5, we obtain the best lower bound by examining all the trees grown in parallel.

5. COMPUTATIONAL RESULTS

The solution methods which were developed and described in the previous sections provide a lower and upper bounds on the value of the optimal Steiner tree solution for a given graph. Optimal solutions were obtained for the problems for which the difference between the value of the upper bound and the lower bound was less than one. In order to investigate the comperative performence of the solution methods developed in this paper, they were programmed and tested on a set of problems taken from Beasley (1984). We present the results of these tests in **TABLE I** below. The algorithms LA1 and LA2 were programmed in FORTRAN and the Dual Ascent Algorithm (D.A. Algorithm in Table I) was programmed in PASCAL. The data set is the one tested in Beasley (1984) and consists of 18 randomly generated problems for undirected graphs. For algorithms LA1 and LA2 we have considered the directed version of these problems by duplicating each arc and assigning directions. The three algorithms were tested using a VAX 8600. In all the tests the number of subgradiant iterations was restricted to 800, the initial δ value was set to 2 and the parameter Idivide was set to 20 for LA1 and to 40 for LA2.

Problem Number	$\|V\|$	$\|A\|$	$\|V_1\|$	Algorithm LA1		Algorithm LA2		D.A. Algorithm	
				Lower Bound	Upper Bound	Lower Bound	Upper Bound	Lower Bound	Upper Bound
1	50	126	9	81.99	82*	81.44	82*	72	83
2	50	126	13	83.00	83*	82.88	90	60	124
3	50	126	25	137.88	138*	137.07	177	107	160
4	50	200	9	59.00	59*	58.95	77	48	93
5	50	200	13	60.98	61*	60.58	65	49	109
6	50	200	25	121.65	122*	121.33	148	84	154
7	75	188	13	110.97	111*	108.79	123	92	138
8	75	188	19	103.99	104*	101.81	118	73	125
9	75	188	38	219.79	220*	207.81	234	207	255
10	75	300	13	85.90	86*	84.81	124	61	175
11	75	300	19	88.00	88*	87.79	127	64	192
12	75	300	38	172.20	174	166.20	228	139	204
13	100	250	17	165.00	165*	162.77	192	94	263
14	100	250	25	234.91	235*	224.56	277	131	310
15	100	250	50	317.60	318*	301.27	353	249	387
16	100	400	17	127.00	127*	122.65	162	73	256
17	100	400	25	128.17	131	124.52	143	101	163
18	100	400	50	215.56	218	209.52	280	182	260

TABLE I : Computational results for the three algorithms for Steiner tree problem on graphs.

Out of the 18 problems attempted, 15 were solved optimally by the LA1 algorithm. Only the first problem was solved optimally by the LA2 algorithm and the Dual Ascent algorithm did not produce a single optimal solution. In terms of the quality of the lower bound values, LA1's lower bound values dominate the values generated by LA2 and the dual ascent algorithm. (The optimal solution is noted by *.)

6. SAMMERY AND CONCLUSIONS

We have presented a number of mathematical formulations for the directed and undirected Steiner tree problem on graphs. These formulations have been used to develop Lagrangean based lower bounding procedures for the problem. In computational tests (on 18 problems used by Beasley (1984, 1987) for testing undirected Steiner tree problems), it has been shown that one of the algorithms (LA1) generates

lower bound values that are close to the optimal solutions. The nonfeasible solutions generated by the Lagrangean based procedure have been incorporated into heuristics which attempt to generate "good" feasible solutions. Here again algorithm LA1 has generated optimal solutions to 15 out of the 18 problems. For the other 3 problems the gap between the feasible and lower bound values were under 2%. Note that in case of an undirected graph, Beasley (1984) reports solving to optimality only 6 out of the 18 problems, and Beasley (1987) again for the undirected graphs reports solving to optimality 15 out of the 18 problems using Cray X-Mp/48 machine. The combined results (Beasley and ours) solve optimally 17 out of the 18 problems.

Based on the above results, it is our believe that algorithm LA1 can be used as an effective tool in Baranch and Bound based procedures for solving the problem.

Acknowledgment : We thank J.E. Beasley for providing us with a copy of his test problems.

REFERENCES

Aneja, Y.P., (1980). "An Integer Linear Programming Approach to the Steiner Problem in Graphs", **Networks**, 10, pp. 167-178.

Beasley, J.E., (1984). "An Algorithm for the Steiner Problem in Graphs", **Networks**, 14, pp. 147-159.

Beasley, J.E., "An SST-Based Algorithm the Steiner Problem in Graphs" , (manuscript March 1987).

Bondy, J.E., Murty, U.S.R., (1976). *Graph Theory with Applications*, North Holland, New York.

Cornuejols, G., Fisher, M.L., and Nemhauser, G.L., (1977). "Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms", **Management Science**, 23, pp. 789-810.

Fisher, M.L., Jaikumar, R., and Van Wassenhove, L., (1986). "A Multiplier Adjustment Method for the Generalized Assignment Problem", **Management Science**, 32, 9, pp. 1095-1103.

Garey, M.R., Johnson, D.S., (1979). *COMPUTERS AND INTRACTABILITY A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco.

Gavish, B., (1978). "On Obtaining the 'best' Multipliers for Lagrangean Relaxation for Integer Programming", **Computers & Operations Research**, 5, pp. 55-71.

Gavish, B., (1982). "Topological Design of Centralized Computer Networks - Formulations and Algorithms", **Networks**, 12, pp. 355-377.

Gavish, B., (1985). "Augmented Lagrangean Based Algorithms for Centralized Network Design", **IEEE Transactions on Communications**, COM-33, 12 .

Gavish, B., and Graves, S.C., (1982). "Scheduling and Routing in Transportation and Distribution Systems: Formulations and New Relaxations", Working Paper, University of Rochester.

Geoffrion, A. M., (1974). "Lagrangian Relaxation and its Uses in Integer Programming", **Math. Programming Study**, Vol. 2, pp. 82-114.

Geoffrion, A.M., and McBride, R., (1979). "Lagrangian Relaxation Applied to Facility Location Problems", **AIIE Transactions**, 10, pp. 40-47.

Hakimi, S.L., (1971). "Steiner's Problem in Graphs and Its Implications", **Networks**, 1, pp. 113-133.

Held, M., Wolfe, P., and Crowder, H.P. (1974). "Validation of subgradient optimization", **Mathematical Programming**, 6, pp. 62-88.

Karp, R.M., (1972)."Reducibility among combinatorial problems", in R.E. Miller and J.W. Thatcher (eds), *Complexity of Computer Computations*, Plenum Press, New York, pp. 85-103.

Magnanti, T.L., and Wong, R.T., (1981). "Accelerating Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria", **Operations Research**, 29, pp. 464-484.

Miller, C.E., Tucker, A.W., and Zemlin, R.A., (1960). "Integer programming formulation of traveling salesman problems", **JACM**, 7, pp. 326-329.

Tarjan, R.E., (1977). "Finding Optimum Branchings", **Networks**, 7, pp. 25-35.

Van Roy, T.J., (1986). "A Cross Decomposition Algorithm For Capacitated Facility Location", **Operations Research**, 34,1, pp. 145-163.

Winter, P., (1987). "Steiner Problem in Networks", **Networks**, 17, pp. 129-167.

Wong, R.T., (1984). "A Dual Ascent Approach For Steiner Tree Problems On A Directed Graph", **Mathematical Programming**, 28, pp. 271-287.